Universal Acceptance Day Uruguay - May 26 , 2025

Theoretical Session: Key Concepts of Universal Acceptance

Nicolás Antoniello – ICANN

Carlos Martínez – LACNIC

Theoretical Session: Key Concepts of Universal Acceptance

Nicolás Antoniello – ICANN
Carlos Martínez – LACNIC

Carlos Martínez: Well, we're here to celebrate a bit, because we heard a series of presentations this morning discussing how great and desirable it is to have a multilingual internet, multi-character domains, etc. Or why someone has to do that,

right? So we 're here to show, I would say, just scratch the surface of the complexity behind it.

I, and I admit that I'm historically a skeptic on the subject, did think the problem was exaggerated until I began to study it a little. The complexity of this is incredible, and we minimize it because in reality, in the language we speak, we have five or six, and perhaps some of the older ones remember the seventh special character, which are the five accents, the ñ and the diaeresis . That's all we have. And the diaeresis , for example, I don't think my son knows what it is.

Other character sets have much, much, much, much worse complexities, such as even character sets where characters are differentiated into narrow and wide versions. The same letter, but it's wide and narrow and means different things. Each of those things has to be encoded in some way.

So, well, today, between this morning and this afternoon, we're going to show you... Talk a little bit about that complexity and show you how to approach and do some things with an email server and a domain. Nico, you 're supposedly the one who knows all this. Go ahead.

Nicolás Antoniello : Well, I've been drinking a little Coca-Cola because my colleagues at ICANN were telling me, don't eat my cornstarch cookies before speaking, let's say, you're going to start... It's complicated, don't eat my cornstarch cookies before speaking. Moral of the day.

Okay. As Cristian said earlier today, ask questions, interrupt, all you want. Again, with these types of presentations, we're supposed to already know what we're presenting. So, these are more for you, to clear up any doubts. And Carlos and I like to do these presentations that have a... By the way, we allow for a large percentage of improvisation. Because I think it makes them more dynamic. But a big part of that also lies in you asking questions and interacting.

The presentation itself will be a guide, although we may not even get to the end of it. And it will depend a bit on how the conversation goes. Yes, it will get a little more technical, just a little, no more. And in the afternoon, it will get a little more technical than now. But the idea is also that everyone can follow along, even if they're not technically inclined. Okay?

Okay. This was said earlier today, several times. So it's going to be a quick review. What is universal acceptance? Basically, supporting writing on all systems,

applications, in other written languages that aren't ASCII code. So, basically, not English. Right? And ASCII code isn't even all of English, so to speak, because there are a ton of English characters, separators, punctuation, etc., that aren't even covered by ASCII code. This isn't English. This is ASCII code, which is even more restrictive than English.

That's the original version of DNS, the Domain Name System. It only supported ASCII characters, ASCII encoding, in domain names. And email too, only ASCII characters for mailbox names. And well, obviously, on the DNS side, what's before the @ sign , which is the mailbox name, and what's after the @ sign . So, to the left of the @ sign , the mailbox name, and to the right of the @ sign, the domain name. Neither of those two, in the original version of those standards, could be written in anything other than ASCII characters. Right?

And I'm going to give you a spoiler . I'm going to give you something that's at the end of the presentation. Because it's not just email. Email is part of what universal acceptance is. In fact, this part of universal acceptance, you'll see an acronym in English, which I think is something like Internationalized Email Address , IEA. IEAS is the email version of universal acceptance, let's say.

But again, it's pointless to have all the email servers accepted. And when I go to ask a user to create a user account and choose a username and password, I tell them to choose, but it's actually a lie. I don't let them choose much because they can only use their email address with the username, and they can choose the password, luckily. But when they go to type in their email address, their email address, if my email address were nicolás at , I don't know, internet.com.uy, I don't think those exist anymore. In fact, mine was nicolás at internet.com.uy without an accent mark, because I couldn't put an accent mark. Now I put the accent mark , but when I go to the web page, it doesn't accept it because the parser , the piece of software, let's say, that was used to see if what I'm putting there is an email, when it sees something that isn't ASCII, it says, "What do you mean, it's not an email?" What do you mean it's not an email? It's a little outdated.

It's about updating all websites and, in general, all applications to support multiple writes, so to speak. Well, that's universally accepted.

Why does it matter? We've also already said it: supporting a diverse and multilingual internet, enabling greater competition, innovation, and consumer choice, creating business opportunities, that's all. From my personal perspective, it's quite relative, but it's also important and one of the drivers of everything in the

world, right? And, well, helping governments develop policies, and also—it's not explicitly stated here, but it's implicit, let's say—there's a lot to maintaining identity, preserving cultural identity, right?

There are people who can't write in any other language. There are people who can't write in any language other than their own and don't know any other languages. And there are others who, by all rights, know three or four or five languages—many more languages than I can possibly know—but they don't want to write in any other language because they want to preserve their identity. So, this is also about that, isn't it?

Okay. So, let's talk a little bit about what universal acceptance is, in a little bit more detail, from the perspective of domain names, that is, from the perspective of the Domain Name System, and what universal acceptance is from the perspective of email addresses, okay?

Raise your hand if you don't know what the Domain Name System is. Don't worry, this isn't a test, so, and you don't necessarily have to know everything about the Domain Name System. Everyone knows what it is. At least everyone thinks they know what it is. And then when you ask them, they say, I know what it is, but I can't explain it. That happens; it happens to all of us. Because knowing how to explain something isn't the same as knowing what something is, let's say. Right? Well, it's supposed to be; if not, it would be, they'd tell me, it's just a matter of assets, and that's it.

Let's see, quickly and just to review , the Domain Name System, or DNS, as we know it by its acronym in English, is that application, which was designed to translate names into addresses, names into unique identifiers for the Internet. The unique identifiers of Internet devices are IP addresses, which are neither the only identifiers, nor are they unique, but well, it's the main characteristic that a device that connects to the Internet must have: it is to have an address, an identifier that differentiates it from all the rest of the devices, and that is the IP address. Right?

And domain names are like the equivalent of people's names in the phone book. Right? The DNS acts as the phone book for the internet. It translates domain names into addresses. Why? Because they're easier to remember than numbers. Yeah, just for that reason. And a whole bunch of other advantages, many of them technical and operational, that came later, but anyway, the main motivation for having a DNS is that: translating names into addresses.

So, now we're going to talk a little bit about how you make a system that originally translated names that could only be written in ASCII into something that can have virtually any type of language, let's say, encoded in a very special way, which Carlos is now going to help me try to explain, let's say, right? Because in UTF-8 it's like, it can take a week to understand. Okay. And talk a little bit about how that translates to... to email.

So, with universal acceptance, it's now possible to have—no, not now, it's been a few years now—domain names and email addresses in any language, right? You have to create, like, a dictionary, right? The coding to be able to support that. Do you want me to... now, or should I cut you off ?

Carlos Martínez: Okay. Well, the problem is this. Everyone's heard... I'm not saying... I'm not saying they know what it is. Have you ever heard of UTF-8? Nobody? Have you? You too? I'm looking at you. Well, here's the thing. Do you know what ASCII is? Or does it sound familiar to you? Does it sound more familiar to you? Okay.

Ultimately, any digital communication system that connects, let's say, that communicates with zeros and ones, and I'm not necessarily referring to computers, in fact , ASCII comes from TELEX, the origin of ASCII comes from TELEX, in the 60s, 70s, 50s, it's a way of representing characters in digital information. The first version of ASCII was 7 bits, it wasn't even 8. Why 7? Because there wasn't even much of a concept that 8 was a reasonable size for a digital computer. And at that time, each bit, not each byte, each bit cost a lot of money. Transmitting a bit appropriately over a long distance cost a lot of... It cost a lot of money.

So, this... An agency in the United States, I think it was the predecessor of the FCC in some ways, said, well, we need a standard way of representing characters so we can send them over TELEX. And they decided on 7 bits because in 7 bits, in 128 combinations, they said, here we can encode everything we need to achieve intelligible communication.

Nico said something very well, that people assume ASCII is for representing English, and that's not the case. ASCII is for representing the smallest subset of English that allows for transmitting something that's more or less understandable. If you search on Google, for example, for images of old telegrams, even in English, you'll see that not even all the characters were transmitted. Often, they omitted the vowels. In words that were just as understandable, they omitted the vowels. Each letter cost a lot of money. So, well, that bit that... It's not like they took it out, it never even

occurred to them to put it in. They said, what's the minimum, minimum, minimum we need? 7. What's the minimum, minimum, minimum we need? Because apart from the characters, lowercase, lowercase, numbers, punctuation, there were certain control sequences. I mean, everything was there. There was actually a strip of bits that was transmitted and everything had to go in there, including the control information, where the message starts, where the message ends, etc. Right .

With the advent of digital computers, there was a transition period where people couldn't quite agree on what was a reasonable bit grouping. There were very strange architectures, 13 bits, 17 bits, 36 bits, things like that. And at some point, people sort of concluded that 8 was a good number. So, that's when the concept of the byte appeared. That's when the concept of the byte appeared.

A few years go by and people say, well, this ASCII we've been using—but we've been using it because it's the only thing there was—couldn't we extend it to 8 bits and use that bit we're not using for something? Okay. And that's where what you're probably already familiar with comes into play: the ASCII you're all familiar with, which is a bit richer than the original ASCII.

Now, as the years go by, personal computing starts to become more popular, Windows comes along. And in Windows, it says, "Microsoft, it would be great to be able to do this multilingually , because we have clients all over the world, but there's no way to represent those characters." And so they invented a hack , literally a hack , which we've been suffering from until now in certain applications, which were what they called Windows Code Pages. Right?

So, what they did was maintain the 7 bits of ASCII as if they were ASCII. But depending on, I don't remember what value, the next 128, or the 128 you gained by adding a bit, they used to represent characters in other languages. So, for example, in Spanish and other Latin languages, it was, I think, CP 1259, something like that. But if you went to Russia, it was another one. And in China, it was another one, etc., etc. And in turn, those 128 you gained weren't enough to represent languages, for example , with Chinese, which has a ton of other symbols. That's clearly a problem, and there was also a standardization effort, which was what led to something called Unicode, which is named somewhere, I think, in the previous slide .

Unicode is about continuing to expand on that idea, representing characters, or more generic things like emojis , for example, and figurines. I have my very personal opinion on that. I don't need that. But, well, continue representing those things as, not as a byte anymore, because one byte isn't enough, but as the

consequences of a byte. But always maintaining that criterion of efficiency I mentioned earlier.

So, simply a simplistic approximation would be, oh well, if one byte isn't enough, I'll add two, or three, or four. The issue is that if each character I'm going to use is going to use four bytes, things really get huge all of a sudden. That may be more or less a problem now, but what happens is that when Unicode started being thought about, storage was still one of the expensive things, and so was transmission cost. Transmission cost, the cost of transmission, is going to become an issue again, which I'll tell you about later.

The issue is this: So these Unicode people said, well, how much space do we need to represent everything? That is, all the characters from all the known languages, plus the smiley face emoji, and the mate emoji, and what have you, and they arrived at a number that I don't know how they arrived at, but it's a kind of strange number, it's one million 111 thousand, I don't know how much, which in Hexa is actually written as 10 FFFF, that's the maximum.

So, well, there you can download the Unicode database, which contains all the mappings of those things. Reading some parts of it is really interesting, because each character comes with what is called its code. point , which is the number that represents it, a description in text, for example the ñ says Latin n with tilde, that is the description they have in English for the ñ, and so on, all, all, all, all, all, that is when I found out about that, that there are wide and narrow characters in some scripts, and they mean different things, that is, the same character, wide and narrow, means different things.

So, in addition to that, once we have those characters represented, the need arises to transmit them through a communication channel. The simplistic approximation would be, how many bits do I need to represent that strange number of a million or something? And roughly, you're fine with four bytes, and it's going to be fine . Since that's super inefficient, the concept of what they call encoding rules arises, which is how to represent that 32-bit number, as efficiently as possible, in bytes, for transmission and storage.

And that's where UTFs come in , there are several UTFs . Unicode Transform Format , there is UTF, Unicode Transform Format . So there's the most common one, which is used in DNS and email, which is UTF-8. Right? It's not because it's what everyone wants, because 8S has to do with the byte; it doesn't really have to do with that. It's simply, in fact , a variable-length encoding.

In UTF-8, ASCII characters are represented. Those that already existed in the original ASCII are represented as their ASCII value. And the other ones, the rare ones, are encoded there, depending on where they are, in that range of 0 to 1,100,000, I don't know how much. They can be, they can be encoded in 1, 2, and 3 bytes, I don't know if there are 4 bytes, maybe there are 4 bytes as well. And there are some bits in the middle that are used to say, well, what comes next, if it's a new character, or if it's a piece of what I'm putting together. So that's the famous UTF-8.

There's UTF-7, which is deprecated; it's no longer used, or shouldn't be used anymore. You always get some surprises there. When you deprecate things, you can be sure that 35 years from now, you're going to open something, and it's going to be encoded in that. And the one that is used, in some applications, is something called UTF-32, which is a very simplistic encoding, which is just that: grabbing each character. The advantage of that is that those who have ever struggled with programming know that processing fixed-length things is generally much easier than processing variable-length things. So, in certain applications, UTF-32 is preferred.

So, this is the first, how can I put it, the first dark secret of all this I'm here to tell you. There it is.

Nicolás Antoniello : And to continue a little with what Carlos was saying, I liked the example of the emojis, it's a very good example for those who know more, are more used to... You know that in many applications, let's say, to put a laughing face, you can write, I don't know, two points, LOL, and that means laughing. out loud , or laughing a lot, let's say, and a colon again. So, they put it between two colons, and whatever's in between, the app interprets it as, oh, I have to encode this, it's an encoding, it's not that I want to convey a colon, LOL, a colon, which doesn't mean anything, but it's a smiling face. So it replaces it with a smiling face.

In the DNS, it's the same, so to speak. Through UTF-8, and an encoding that for the DNS, for the sake of universal acceptance, is called Puny. Code is written using ASCII characters, something that applications and systems then interpret as what it should be, which is an internationalized character, meaning, internationalized means anything that is not ASCII, let's say, right?

Well, the truth is that I'm thinking it's a little bit, well, like, it doesn't matter, internationalized when it's not ASCII, it's like when we say, I'm going inland, and suddenly I go to Piriápolis, Piriápolis is no more, there's nothing further out than

Piriápolis, because it's on the coast, but for us it's inland, right? Well, this is the same. Internationalized means it's not ASCII, okay, let's say that .

Well, then, and that's how it's encoded, in the DNS and that's how it's encoded in the , in the , in the email, and it's a little bit, a little bit also what, what that MIME format does, which, which allows you to even take an image and convert it into text and transmit it as text and then on the other side it's reconstructed and it's an image, let's say, right? So, to send a little image like that, I generate 57 sheets of, of MIME, of text, and then that, again, um, in an absolutely inefficient way, unless I compress it, I, I translate it back into the image.

So, going a little bit into email and getting more into the topic of email, as a review, these are like three big parts of an email: the envelope, the header of the message, and the body of the message. Email, when it was created, was in electronic format, traditional postal mail, right? So, the virtual envelope, in the case of email, has a sender and a recipient on the outside, let's say. Those are part of the message headers, and they are used by the delivery system and, how do you say it? Delivery would be like, yes, like sending. The email delivery system uses those headers and that information to make the email, right? Uh, for example, from an email client where I compose an email to the recipient's mailbox, going through a bunch of servers and devices in between.

And the body of the message is what goes inside the envelope, right? I'm telling you, universal acceptance isn't about getting into the body of the message. If the body of the message is written in whatever language it's written in, the, the, the, the universal acceptance project doesn't care because that's what goes inside the envelope. Just like the postman, quote, doesn't care about what goes inside the envelope, right?

To be honest, that was a problem at one time. But that was resolved quite a while back when, in general, almost all operating systems, more or less in a time window around the late 1990s, adopted UTF-8. All operating systems became UTF-8. So, what goes into email, or a Word document, or even a text file, stopped being a problem. File names, for example, remained a problem for a long time, and well, and this continues to be a problem to this day.

Which doesn't mean it's not a problem for security reasons, etc., etc. But, well, that's the same thing. I can send a bomb in an envelope, and I can send a compressed file in an email that's a digital bomb, let's say, right? Some kind of attack, a Trojan horse virus. Nowadays, you have an endless array of damages that

can be done with an email, right? But, well, for the purposes of universal acceptance, we're not going to focus on what's inside, but rather on what's outside the envelope. Right?

Okay. These are definitions that, when you cross the threshold, if you forget them, your life will be better. They're like the different devices and agents involved in the chain of sending, receiving, and transmitting an email, right? Then there's the MUA, which is nothing more than a very complicated acronym to describe the email client, which can be any email client, I don't know, Outlook, what's the name of Apple's? I forget now, but anyway, there are tons, whatever you want.

At the time, here I am looking at Martín who worked, in another life we worked together somewhere, we used Paint, do you remember Paint? A text-based email client that was something just for nerds, let's say. Yeah, yeah, everything was done with keys like that. Huh? It wasn't the worst. And it wasn't the worst, there were worse, there were worse, there were worse. You had high... Yeah, yeah, yeah, you did things much faster today and for some, um, I wish I had continued that, that... For the Paint nostalgics continue existing the MUT. Well, we're going to use it this afternoon for a demonstration, so there you have it.

This, the MSA, which is already on the server side, let's say, this, the email, actually, in the , in the , is part of the system that sends the email that one generates in the MUA, let's say. The MSA is the one that initiates the sending, the first one to send the generated email, let's say. Then, MTAs are all the, let's put it bluntly and quickly, let's say, but to be clear, they are all the systems through which the email passes. All the intermediate servers through which an email passes when it travels from the origin to a destination.

So, you'll see, typically, there's going to be an MUA, an MSA, several MTAs , and an MDA at the end, which is the one that receives the email, and the MDA is the program that takes the incoming email and puts it in the inbox, deposits it in the recipient user's inbox, let's say. Right?

Okay. Why are we talking about all that anyway ? Then there came the question: it wasn't IEA, it's EAI. Okay. Having UTF support, things that have to happen, which Carlos already explained a little bit, you have to have UTF-8 support to be able to generate these encodings in languages other than ASCII, both for the mailbox name and the domain name.

universalized domain support, my, my entire DNS system has to support internationalized domains and this means not only that my DNS server where I put the information that are called authoritative servers have support for this, but the recursive servers have to have it as well , I have a lot of, all the servers that participate in the resolution of the domain name have to support that.

And the same thing, if I want to have an internationalized mailbox name, even if I don't have an internationalized domain, for example, nicolascontildela@internet.com.uy or whatever, I'm going to need all the servers involved in email—that is, this whole family of servers we were talking about—every one of them, wherever that email passes through, to support it. If the email reaches an intermediate server, an intermediate MTA, for example, that doesn't support that, it's dead. That's as far as it goes. Right?

And what happens after that is like a roulette wheel. There are some systems—this is a bit of a personal opinion —that are well-configured, that send a response back so the originator knows the reason why I can't get past that. Basically, whoever rejects it will say, "I can't stand universalization of mailboxes or domain names or neither." The email has reached this point, but I'm warning you that's why I 'm going to discard it. If I discard it, bye. And that email will never reach its destination, let's say. No matter how many times I send it. So, the entire chain has to support universal acceptance. That's one of the obstacles to developing it. It doesn't depend only on me; it depends on everything in between.

Well, there are some things that aren't universally accepted. Let's say, UTF-8 already existed long before and was used. It's another tool that was also used for this issue. The content of the email subject isn't of interest for sending the email. In fact, it's not of interest; it's only of interest to the sender and the recipient. Address comments. We won't go into detail about this, but anyway, the address is repeated twice. It goes outside the envelope for the mail carrier to use, and it goes inside the envelope again. The one inside the envelope doesn't matter. The one inside the envelope could even lie about what's inside the envelope, but what matters to the mail carrier is what's outside the envelope, and that's universally accepted as well. And obviously the body of the message isn't either, because it's inside the envelope, and I can write that in any language I want.

And well, for the universalization of all that isn't exactly universally accepted, MIME already exists, which, combined with UTF-8, allows you to represent almost anything. For example, in the body of an email message, even an image, a word processor document, a PowerPoint presentation, a PDF, anything.

Okay. So, going into a little more detail and configuration, because we're almost at the hour mark, but we're going to go for five more minutes. We'll leave some requirements for this presentation so that those who are more technically inclined, those who are more curious, can have them detailed. These are some issues to keep in mind when I'm configuring universal acceptance or when I want to provide universal acceptance support.

Well, that's like a little recipe, you have to normalize the Unicode string before processing it in UTF-8, use the correct format for IDN, and well, there are several formats for many of these standards that were used at the beginning, but they are no longer used. It is not recommended to use them anymore, but rather it is recommended to always use the latest version, let's say, of all these things. If, for example, the IDN , many applications are programmed using this version, version 2003, this version is no longer recommended to use, so now those who already have this programmed will have to release a patch with this version, and those who are just starting to program should not use this one, use that version of IDN.

: Did you explain what IDNA is ? No, tell them . Have you realized that this Unicode thing lets you do a lot of things and represent a lot of things? In reality, it lets you do so many things and represent so many things that it becomes dangerous, in a certain sense. So, for the purposes of being used on the Internet, for Internet communications, not all the variants, the enormous variety of things that Unicode lets you represent are allowed. So there are a series of rules that determine what things are valid and what things aren't valid for sending over the Internet. That would be valid within the body of the message, a bit like what Nico was saying, suddenly.

The most obvious, for example, is that emojis aren't allowed in top-level domains . I mean, there will never be—or, well, as long as we're alive, there won't be, over my dead body, there will never be a top-level domain that's dot mate, let's say. Dot mate, little dot mate, I mean. That's a funny example, but there are many such things. Because there's a whole security issue linked to this that we might be able to discuss, which is homograph attacks. Which is tremendous.

I mean, the amount of things you can do and the amount of deception you can commit in a world where people call you on the phone and ask for money, and you give it to them. I mean, imagine the things you can do by drawing with strange characters. It's incredible. So, well, there are a series of rules that dictate what things are valid to represent in email addresses, domain names, especially domain

names, which have to do with those IDNA rules. In addition to some coding rules additional and such .

Nicolás Antoniello : Okay. I left the presentation until the end because we'll be covering the rest of the presentation in the afternoon as we work through the demo lab. We're going to do a sort of demo with Carlos Guiada in the afternoon, after lunch. And then we'll be covering the presentation.

Acceptance website. Steering Group , let's say, which can be used to check if your email server supports email internationalization. That is, if it basically supports universal acceptance. Right? And they go there. And basically, they'll generate an email, send it, and the system will see if it gets a response, and then what level of response it gets.

Because within the issue of... of universal acceptance and support for systems... Sorry, email... This is the problem with eating... Ah, cornstarch before... Email has several levels of support. And it doesn't mean you have to comply with all of them. In other words, you have to have absolute and complete support to be able to transmit an email. For example, if it's an MTA, which is an intermediate agent, which is neither the one that generates the email nor the recipient and doesn't support creating a universalized mailbox... Well, it doesn't matter, because the MTA isn't actually where the mailbox is created or where that is managed, but rather the MTA has to receive an email and pass it on. It would be like a kind of intermediate mail agency.

But it does have to support that. It has to understand that it's an internationalized character in the name of a mailbox or domain in order to forward email to its destination. So there are different levels. So, with this tool, you can check if and what level I support it. And there are several types. Type 1, type... In general, it's type 1 support, type 1 support, and type 2 support, let's say. Type 2 is the one that supports everything. Type 1 is partial support that can be used, for example, in an MTA, but it can't be used for either an email client agent or an email recipient, let's say.

Well, we already talked about this a little bit at the beginning. Beyond email, the idea is to... Like... Invite them to... Invite us all to that, to review not only the software itself, but at the application level, at the web user interface level, whether or not we're supporting universal acceptance.

In fact, just the software is not enough, because if we forget about all the forms, for example, to create an email address, if the interface of... Today there are, for example, all the DNS servers that are used today, authoritative and recursive, in all their versions, in all their flavors, let's say, from any manufacturer, in versions from, I don't know, five years ago to date, they support universalization of domain names.

You'd be surprised at the extremely small number of registrars, agents who market domain names, that support universal acceptance, that support universalization. They're very, very few. All their DNS systems already support it. What doesn't support it is the web interface, the billing systems, or the processing systems. So you also have to make an investment in that, which I think is the biggest one, because the rest is just updating the software and that's it.

Of course, like everything else, you have to tell them one thing, do another. You have to train people, you have to train the operations teams, etc. But that's also the point of these meetings, to at least kick things off so you can begin exploring this. Well, that's it for now.

Moderator: Well, thank you very much , Carlos and Nicolás. I didn't mention it, but Carlos and Nicolás are obviously members of the chapter, founding members, and Carlos was also the president of ISOC-Uruguay, which I forgot to mention. But, well, we all know them, and I told you they were a good duo giving presentations. Now, continuing with this topic, speaking of the emojis you mentioned, Carlos, there are several discussion groups in the GNSO at ICANN about the inclusion of emojis.

Carlos Martínez: Tempting, I mean, it opens up a ton of business opportunities. Many. Not very genuine, but, well, business nonetheless. But yes, it's certainly a topic of discussion, but it also creates a whole world of gigantic problems.

Moderator: Yes, there were precisely those in favor, those against, several very interesting discussions about it.